Application 09/470,566

#8
Rose
1-22-03

CLEAN COPY OF SPECIFICATION AND CLAIMS AS AMENDED

Patent Application of
Kendyl A. Román
Cyrus J. Hoomani
and
Richard S. Neale
for

RECEIVED
JAN 21 2003
Technology Center 2600

TITLE: GENERAL PURPOSE COMPRESSION FOR VIDEO IMAGES (RHN)

CROSS-REFERENCE TO RELATED APPLICATIONS

This application claims priority under 35 U.S.C. § 119(e) of the co-pending U.S. provisional application Serial Number 60/113,276 filed on 1998 December 23, and entitled "METHOD OF IMAGE ENHANCEMENT, COMPRESSION, AND ENCODING of GRAYSCALE IMAGES (ECHOCODEC)." The provisional application Serial Number 60/113,276 filed on 1998 December 23 and entitled "METHOD OF IMAGE ENHANCEMENT, COMPRESSION, AND ENCODING of GRAYSCALE IMAGES (ECHOCODEC)" is also hereby incorporated by reference.

Our co-pending U.S. patent application Serial Number 90/312,922 filed on 1999 May 17, and entitled "SYSTEM FOR TRANSMITTING VIDEO IMAGES OVER A COMPUTER NETWORK TO A REMOTE RECEIVER" describes an embodiment of the invention of this application in combination with our co-pending U.S. provisional application Serial Number 60/085,818 filed on 1998 May 18, and entitled "APPARATUS FOR TRANSMITTING LIVE VIDEO IMAGES OVER A COMPUTER NETWORK TO MULTIPLE REMOTE RECEIVERS." U.S. patent application Serial Number 90/312,922 filed on 1999 May 17, and entitled "SYSTEM FOR TRANSMITTING VIDEO IMAGES OVER A COMPUTER NETWORK TO A REMOTE RECEIVER" is also hereby incorporated by reference.

BACKGROUND – FIELD OF THE INVENTION

This invention relates to data compression, specifically to the compression and decompression of video images.

BACKGROUND–DESCRIPTION OF PRIOR ART

In the last few years, there have been tremendous advances in the speed of computer processors and in the availability of bandwidth of worldwide computer networks such as the Internet. These advances have led to a point where businesses and households now commonly have both the computing power and network connectivity necessary to have point-to-point digital communications of audio, rich graphical images, and video. However the transmission of video signals with the full resolution and quality of television is still out of reach. In order to achieve an acceptable level of video quality, the video signal must be compressed significantly without losing either spatial or temporal quality.

A number of different approaches have been taken but each has resulted in less than acceptable results. These approaches and their disadvantages are disclosed by Mark Nelson in a book entitled <u>The Data Compression Book, Second Edition</u>, published by M&T Book in 1996. Mark Morrision also discusses the state of the art in a book entitled <u>The Magic of Image Processing</u>, published by Sams Publishing in 1993.

Video Signals

Standard video signals are analog in nature. In the United States, television signals contain 525 scan lines of which 480 lines are visible on most televisions. The video signal represents a continuous stream of still images, also known a frames, that are fully scanned, transmitted and displayed at a rate of 30 frames per second. This frame rate is considered full motion. A television screen has a 4:3 aspect ratio.

When an analog video signal is digitized, each of the 480 lines are sampled 640 times and each sample is represented by a number. Each sample point is called a picture element, or pixel. A two dimensional array is created that is 640 pixels wide and 480 pixels high. This 640 x 480 pixel array is a still graphical image that is considered to be full frame. The human eye can perceive 16.7 thousand colors. A pixel value comprised of 24 bits can represent each perceivable color. A graphical image made up of 24-bit pixels is considered to be full color. A single second-long full frame, full color video requires over 220 millions bits of data.

The transmission of 640 x 480 pixels x 24 bits per pixel times 30 frames requires the transmission of 221,184,000 millions bits per second. A T1 Internet connection can transfer up to 1.54 millions bits per second. A high speed (56Kb) modem can transfer data at a maximum rate of 56 thousand bits per second. The transfer of full motion, full frame, full color digital video over a T1 Internet connection, or 56Kb modem, will require an effective data compression of over 144:1, or 3949:1, respectively.

Basic Run-length Encoding

An early technique for data compression is run-length encoding where a repeated series of items are replaced with one sample item and a count for the number of times the sample repeats. Prior art shows run-length encoding of both individual bits and bytes. These simple approaches by themselves have failed to achieve the necessary compression ratios.

Variable Length Encoding

In the late 1940s, Claude Shannon at Bell Labs and R.M. Fano at MIT pioneered the field of data compression. Their work resulted in a technique of using variable length codes where codes with low probabilities have more bits, and codes with higher probabilities have fewer bits. This approach requires multiple passes through the data to determine code probability and then to encode the data. This approach also has failed to achieve the necessary compression ratios.

D. A. Huffman disclosed a more efficient approach of variable length encoding known as Huffman coding in a paper entitled “A Method for Construction of Minimum Redundancy Codes,” published in 1952. This approach also has failed to achieve the necessary compression ratios.

Arithmetic, Finite Context, and Adaptive Coding

In the 1980s, arithmetic, finite coding, and adaptive coding have provided a slight improvement over the earlier methods. These approaches require extensive computer processing and have failed to achieve the necessary compression ratios.

Dictionary-Based Compression

Dictionary-based compression uses a completely different method to compress data. Variable length strings of symbols are encoded as single tokens. The tokens form an index to a dictionary. In 1977, Abraham Lempel and Jacob Ziv published a paper entitled, "A Universal Algorithm for Sequential Data Compression" in IEEE Transactions on Information Theory, which disclosed a compression technique commonly known as LZ77. The same authors published a 1978 sequel entitled, "Compression of Individual Sequences via Variable-Rate Coding," which disclosed a compression technique commonly known as LZ78 (see U.S. Patent 4,464,650). Terry Welch published an article entitled, "A Technique for High-Performance Data Compression," in the June 1984 issue of IEEE Computer, which disclosed an algorithm commonly known as LZW, which is the basis for the GIF algorithm (see US Patents 4,558,302, 4,814,746, and 4,876,541). In 1989, Stack Electronics implemented a LZ77 based method called QIC-122 (see U.S. Patent 5,532,694, U.S. Patent 5,506,580, and U.S. Patent 5,463,390). The output of a QIC-122 encoder consists of a stream of data, which, in turn consists of tokens and symbols freely intermixed. Each token or symbol is prefixed by a single bit flag that indicates whether the following is a dictionary reference or a plain symbol. The definitions for these two sequences are:

(a) plaintext: <1><eight-bit-symbol>

(b) dictionary reference: <0><window-offset><phrase-length>

Windows offsets are encoded as seven bits or eleven bits. These lossless (method where no data is lost) compression methods can achieve up to 10:1 compression ratios on graphic images typical of a video image. While these dictionary-based algorithms are popular, these approaches require extensive computer processing and have failed to achieve the necessary compression ratios.

JPEG and MPEG

Graphical images have an advantage over conventional computer data files: they can be slightly modified during the compression/decompression cycle without affecting the perceived quality on the part of the viewer. By allowing some loss of data, compression ratios of 25:1 have been achieved without major degradation of the perceived image. The Joint Photographic Experts Group (JPEG) has developed a standard for graphical image compression. The JPEG lossy (method where some data is lost) compression algorithm first divides the color image into three color planes and divides each plane into 8 by 8 blocks, and then the algorithm operates in three successive stages:

(a) A mathematical transformation known as Discrete Cosine Transform (DCT) takes a set of points from the spatial domain and transforms them into an identical representation in the frequency domain.

(b) A lossy quantization is performed using a quantization matrix to reduce the precision of the coefficients.

(c) The zero values are encoded in a zig-zag sequence (see Nelson, pp. 341-342).

JPEG can be scaled to perform higher compression ratio by allowing more loss in the quantization stage of the compression. However this loss results in certain blocks of the image being compressed such that areas of the image have a blocky appearance and the edges of the 8 by 8 blocks become apparent because they no longer match the colors of their adjacent blocks. Another disadvantage of JPEG is smearing. The true edges in an image get blurred due to the lossy compression method.

The Moving Pictures Expert Group (MPEG) uses a combination of JPEG based techniques combined with forward and reverse temporal differencing. MPEG compares adjacent frames and for those blocks that are identical to those in a previous or subsequent frame and only a description of the previous or subsequent identical block is encoded. MPEG suffers from the same blocking and smearing problems as JPEG.

These approaches require extensive computer processing and have failed to achieve the necessary compression ratios without unacceptable loss of image quality and artificially induced distortion.

QuickTime: CinePak, Sorensen, H.263

Apple Computer, Inc. released a component architecture for digital video compression and decompression, named QuickTime. Any number of methods can be encoded into a QuickTime compressor/decompressor (codec). Some popular codec are CinePak, Sorensen, and H.263. CinePak and Sorensen both require extensive computer processing to prepare a digital video sequence for playback in real time; neither can be used for live compression. H.263 compresses in real time but does so by sacrificing image quality resulting in severe blocking and smearing.

Fractal and Wavelet Compression

Extremely high compression ratios are achievable with fractal and wavelet compression algorithms. These approaches require extensive computer processing and generally cannot be completed in real time.

SUMMARY OF THE INVENTION

In accordance with the present invention a method of compression of a video stream comprises steps of sub-sampling a video frame, determining a code for each pixel, run-length encoding the codes whereby the method can be executed in real time and the compressed

representation of codes saves substantial space on a storage medium and require substantially less time and bandwidth to be transported over a communications link. The present invention includes a corresponding method for decompressing the encoded data.

Objects and Advantages

Accordingly, beside the objects and advantages of the method described in our patent above, some additional objects and advantages of the present invention are:

(a) to provide a method of compressing and decompressing video signals so that the video information can be transported across a digital communications channel in real time.

(b) to provide a method of compressing and decompressing video signals such that compression can be accomplished with software on commercially available computers without the need for additional hardware for either compression or decompression.

(c) to provide a high quality video image without the blocking and smearing defects associated with prior art lossy methods.

(d) to provide a high quality video image that suitable for use in medical applications.

(e) to provide some level of encryption so that images are not directly viewable from the data as contained in the transmission.

(f) to provide a method of compression of video signals such that the compressed representation of the video signals is substantially reduced in size for storage on a storage medium.

(g) to enhance electronically generated images by filtering highs and lows.

DRAWING FIGURES

In the drawings, closely related figures have the same number but different alphabetic suffixes.

Fig 1 shows the high level steps of compression and decompression of an image.

Fig 2A to 2H show alternatives for selecting a pixel value for encoding.

Fig 3A shows the encode table of the preferred embodiment.

Fig 3B shows a chart of values corresponding to the sample encode table.

Fig 4A shows the flowchart for the preferred embodiment of the compression method.

Fig 4B shows an image and a corresponding stream of pixels.

Fig 5A to 5C shows the formats for the run-length encoding.

Fig 6 shows a series of codes and the resulting encoded stream.

Fig 7 shows a sample decode table.

Fig 8 shows an alternate method of selecting a five bit code.

Fig 9 shows the flow chart for the preferred embodiment of the decompression method.

Figs 10A to 10C show an encryption key, an encryption table and a decryption table.

Reference Numerals in Drawings

100	compression steps	110	sub-sampling step
120	code lookup step	130	run-length encoding step
140	encoded data	150	decompression steps
160	run-length expansion step	170	value lookup step
180	image reconstitution step	200	32 bit pixel value
202	blue channel	204	green channel
206	red channel	208	alpha channel
210	24 bit pixel value	212	blue component
214	green component	216	red component
220	RGB averaging diagram	222	blue value
224	green value	226	red value
228	averaged value	230	blue selection diagram
232	blue instance	234	green instance
236	red instance	240	selected blue value
250	green selection diagram	260	selected green value
270	red selection diagram	280	selected red value
290	grayscale pixel	292	grayscale blue
294	grayscale green	296	grayscale red
298	selected grayscale value	299	8 bit pixel value
300	encode table		
310	codes	320	line comments
330	minimum values	340	maximum values
360	stepped values	350	chart of values
370	line numbers	400	encode flowchart
402	encode entry	403	encode initialization step
404	get pixel step	405	get value step
406	lookup encoded value step	408	compare previous
410	increment counter step	412	check count overflow

414	new code step	416	check end of data
418	set done	420	counter overflow step
422	check done	428	encode exit
430	image	440	image width
450	image height	460	pixel stream
500	code byte	510	flag bit
520	repeat code	530	count
550	data code	565	data bit 6
570	data bit 5	575	data bit 4
580	data bit 3	585	data bit 2
590	data bit 1	595	data bit 0
610	decimal values	620	first value
622	second value	624	third value
626	fourth value	628	fifth value
630	sixth value	632	seventh value
640	binary code	650	first byte
651	repeat count	652	second byte
653	first code	654	third byte
655	second code	656	fourth byte
657	third code	700	decode table
710	alpha values	720	red values
730	green values	740	blue values
800	8 bit pixel	810	pixel bit 7
812	pixel bit 6	814	pixel bit 5
816	pixel bit 4	818	pixel bit 3
820	pixel bit 2	822	pixel bit 1
824	pixel bit 0	830	5 bit sample
832	sample bit 4	834	sample bit 3
836	sample bit 2	838	sample bit 1
840	sample bit 0	850	ignored bits
860	decompressed pixel	880	3 low order bits
900	decode entry	901	decode initialize step
902	get code step	906	determine type

908	decode lookup step	909	check zero count
910	place pixel step	912	assign counter step
914	reset counter step	916	check length
918	decode exit	1000	encryption key
1010	encryption table	1020	decryption table

Terminology Correlation

Different terminology was used in the specification of application 09/312,922. The two specifications were written by two different authors who described distinct embodiments of the subject invention with two distinct sets of terminology. The following table provides a partial correlation between the two sets of terminology. Note that this correlation of terms does not necessarily mean the terms are equivalent. This high level correlation is provided to aid in understanding similarities and differences between the two specifications. Also note that the same term used in this specification in the field of "compression and decompression of video images" is likely to have different meaning in the 09/312,922 specification which was written in the field of "video communications systems" and "medical devices". For example in the field of compression the term "image" generally refers to a still image or a single frame of a video, but in the medical device field the term "image" often refers to the collection of all the frames in a video. Thus the terms themselves should be understood based on the respective original specification. The legal presumption that terms in related applications by the same inventors have the same meaning does not apply in this case.

Terms Used In This Application		Terms Used in Application 09/312,922
140	encoded data	stream of compressed data
212	blue component	blue scale value
214	green component	green scale value
216	red component	red scale value
300	encode table, encodeTable	encodePallet
	decodeTable	decodePallet
310	codes	compression lookup table 310
	code	line number
320	line comments	documentation 320
350	chart of values	lookup table 350
370	line numbers	line number (column)

430	image	video image 400
460	pixel stream	stream of pixel data 405
500	code byte	data structure 500
510	flag bit	identification bit 510
520	repeat code	repeat data structure 525
530	count	repeat value
530	count	repeat counter value
530	count	remaining bits 510
550	data code	line number data structure 550
610	decimal values	uncompressed data stream 1000
640	binary code	compressed data stream 1020
700	decode table	decompression lookup table 700
720	red values	red scale illumination intensity value 710
730	green values	green scale illumination intensity value 720
740	blue values	blue scale illumination intensity value 730
	machine	system

DESCRIPTION OF THE INVENTION

Fig 1–Compression and Decompression Steps

Fig 1 illustrates a sequence of compression steps 100 and a sequence of decompression steps 150 of the present invention. The compression steps 100 comprise a sub-sampling step 110, a code lookup step 120 and a run-length encoding step 130. After completion of the compression steps 100, a stream of encoded data 140 is output to either a storage medium or a transmission channel. The decompression steps 150 comprise a run-length expansion step 160 wherein the stream of encoded data 140 is processed, a value lookup step 170 and an image reconstitution step 180.

Figs 2A to 2H Selecting Pixel Values for Encoding

Figs 2A to 2G illustrate alternatives for selecting a pixel value for encoding. The sub-sampling step 110 (Fig 1) includes sub-sampling of an 8 bit pixel value to obtain a value for use in the subsequent code lookup step 120 (Fig 1).

Video digitizing hardware typical has the options of storing the pixel values as a 32 bit pixel value 200 or a 24 bit pixel value 210, shown in Fig 2A and Fig 2B, respectively. The 32 bit pixel value 200 is composed of a blue channel 202, a green channel 204, a red channel 206, and an alpha channel 208. Each channel contains of 8 bits and can represent 256 saturation levels for the

particular color channel. For each channel the saturation intensity value of zero represents the fully off state, and the saturation intensity value of "255" represents the fully on state. The 24 bit pixel value 210 is composed of a blue component 212, a green component 214, and a red component 216. There is no component for the alpha channel in the 24 bit pixel value 210. Regardless of the structure, the blue channel 202 is equivalent to the blue component 212, the green channel 204 is equivalent to the green component 214, and the red channel 206 is equivalent to the red component 216.

In the present invention, the 32 bit pixel value 200 alternative is preferred due to the consistent alignment of 32 bit values in most computer memories; however for simplicity of illustration the alpha channel 208 will be omitted in Fig 2C to 2G.

If the video signal is digitized in color, the three color components may have different values. For example in Fig 2C, a RGB averaging diagram 220 illustrates a blue value 222 of 35 decimal, a green value 224 of 15, and a red value 226 of 10. One alternative is to sub sample from 24 bits to 8 bits by averaging the three color values to obtain an averaged value 228 that, in this example, has the value of 20. $(10+15+35)/3 = 20$

Fig 2D illustrates another alternative for selecting an 8 bit value in a blue selection diagram 230. In this example, a blue instance 232 has the value of 35, a green instance 234 has the value of 15, and a red instance 236 has the value of 10. In this alternative the blue instance 232 is always selected as a selected blue value 240.

Fig 2E illustrates another alternative for selecting an 8 bit value in a green selection diagram 250. In this alternative the green instance 234 is always selected as a selected green value 260.

Fig 2F illustrates another alternative for selecting an 8 bit value in a red selection diagram 270. In this alternative the red instance 236 is always selected as a selected red value 280.

If the video signal being digitized is grayscale, the three color components will have the same values. For example, in Fig 2G, a grayscale pixel 290 comprises a grayscale blue 292 with a value of decimal 40, a grayscale green 294 with a value of 40, and a grayscale red with a value of 40. Because the values are all the same, it makes no difference which grayscale color component is selected, a selected grayscale value 298 will have the value of 40 in this example.

The preferred embodiment of this invention uses the low order byte of the pixel value, which is typically the blue component as shown in Fig 2D.

Fig 2H illustrates an 8 bit pixel value 299 which is selected by one of the alternatives described above. The 8 bit pixel value 299 is equivalent to items referenced by numerals 228, 240, 260, 280, or 298. This reduction of the 32 bit pixel value 200 or the 24 bit pixel value 210

contributes a reduction in data size of 4:1 or 3:1, respectively. This reduction recognizes that for some images, such as medical images or grayscale images, no relevant information is lost.

Fig 3A–Encode Table

Fig 3A illustrates an encode table 300 of the preferred embodiment of the present invention. The encode table 300 may be implemented in the C programming language, as shown, and is an array of 256 bytes. These bytes are the codes 310. Each line of the array elements is documented with one of a series of line comments 320. The line comments 320 contain a column of minimum values 330, a column of maximum values 340, and a column of stepped values 360. The first column of encode table 300 is a column of line numbers 370. The encode table is arranged such that each line contains codes 310 that have a value equal to its line number.

The encode table 300 reduces the 8 bit value 299 to a 5 bit code. This reduction recognizes that for some images, such as medical images, no relevant information was lost. This reduction also eliminates noise from the video signal and thereby increases the efficiency of the run-length encoding step 130 (Fig 1).

Fig 3B–Chart of Values

Fig 3B, a chart of values 350, enumerates the range that the 8 bit pixel value 299 can have and the respective placement of each value in the encode table 300. The first column of the chart enumerates the line numbers 370. Column A contains the minimum values 330 for each line. Each row of the chart has from four to nine sequential entries. The last column of the chart, column J, contains the stepped values (360) that are used in the value lookup step 170 (Fig 1).

Fig 4A–Encode Flowchart

Fig 4A illustrates the encode flowchart 400 which represents the details of the preferred embodiment of the code lookup step 120 (Fig 1) and the run-length encoding step 130 (Fig 1) for the present invention.

The encoding begins at an encode entry 402. In a encode initialization step 403, a prior value P is set to a known value, preferably decimal "255" or hexadecimal 0xFF, a repeat count C is set to zero, an encoded length L is set to 0, and a completion flag "Done" is set to a logical value of false. Next, a get pixel step 404 obtains a pixel from the image being encoded. At a get value step 405, a value V is set to the 8 bit pixel 299 as derived from the pixel using one of the methods shown in Fig 2C to 2G, preferably the fastest as explained above. At a lookup encoded value step 406, an encoded value E is set to the value of one of the codes 310 (Fig 3) of the encode table 300 as indexed by V. Next, a compare previous 408 decision is made by comparing the values of E and P. If the

values are the same, a increment counter step 410 is executed and flow continues to the get pixel step 404 that obtains the next pixel from the image.

If the encode value E does not match the prior value P, then a check count overflow 412 decision is made. If the counter C is less than or equal to 128, then a new code step 414 is executed, otherwise a counter overflow step 420 is executed.

At step 414, the counter C is bit-wise AND-ed with hexadecimal 0x80 which sets the high order bit to a binary value of 1 and is placed in the encoded data 140 buffer A at the next available location as indexed by the encoded length L. Then, continuing inside flowchart step 414, L is incremented, the prior value P is placed in the encoded data 140 buffer A, L is incremented, the repeat count C is set to 1 and the prior value P is set to the encode value E. After step 414, a check end of data decision is made by checking to see if there are any more pixels in the image and if not if the last value is has been processed. Because this method utilizes a read ahead technique, step 414 must be execute one more time after the end of data is reached to process the last run-length. If there is more data in the image, flow continues to a check of the completion flag "Done" at step 422. If the check indicates that the process is not completed, flow continues to step 404.

If the end of data is reached but the completion flag "Done" is still false, flow continues to a set done step 418. At step 418, the completion flag "Done" is set to logical true, and flow continues to decision 412 where the last run-length will be output and flow will eventually exit through step 414, decision 416, decision 422, and then terminate at encode exit 428.

It is possible for the repeat count C to become larger than 128 requiring more bits than allocated by this method. This situation is handled by making the check count overflow 412 decision and executing the counter overflow step 420. At step 420, the value hexadecimal 0x80 is placed in the encoded data 140 buffer A at the next available location as indexed by the encoded length L. Then, continuing inside flowchart step 420, L is incremented, the prior value P is placed in the encoded data 140 buffer A, L is incremented, the repeat count C is decrement by 128. After step 420, flows continues to the check count overflow 412 decision. Thus when the encode value E repeats more that 128 times, multiple sets of repeat counts and encoded values are output to the encoded data 140 buffer.

This entire process is repeated for each image or video frame and the encoded length L is transmitted with the encoded data associated with each frame. The encoded length varies from frame to frame depending on the content of the image being encoded.

Fig 4B–Image and Pixel Stream

Fig 4B illustrates an image and its corresponding stream of pixels. A rectangular image 430 is composed of rows and columns of pixels. The image 430 has a width 440 and a height 450, both measured in pixels. Pixels in a row are accessed from left to right. Rows are accessed from top to bottom. Some pixels in the image are labeled from A to Z. Pixel A is the first pixel and pixel Z is the last pixel. Scanning left to right and top to bottom will produce a pixel stream 460. In the pixel stream 460, pixels A and B are adjacent. Also pixels N and O are adjacent even though they appear on different rows in the image. If adjacent pixels have the same code the process in Fig 4A will consider them in the same run.

Because the video signal being digitized is analog there will be some loss of information in the analog to digital conversion. The video digitizing hardware can be configured to sample the analog data into the image 430 with almost any width 440 and any height 450. The present invention achieves most of its effective compression by sub-sampling the data image with the width 440 value less than the conventional 640 and the height 450 value less than the convention 480. The preferred embodiment of the invention for use in a medical application with T1 Internet transmission bandwidth is to sample at 320 by 240. However, a sampling resolution of 80 by 60 may be suitable for some video application.

Figs 5A to 5C–Run-length Encoding Formats

Figs 5A to 5C show the formats for the run-length encoding. Fig 5A shows a code byte 500, with its high order bit designated as a flag bit 510.

Fig 5B shows a repeat code 520 comprising a Boolean value one in its flag bit 510 and a 7 bit count 530 in the remaining 7 low order bits. The seven bit count 530 can represent 128 values with a zero representing "128" and 1 through 127 being their own value.

Fig 5C shows a data code 550 comprising:

1. a Boolean value zero in its flag bit 510
2. two unused data bits: data bit 6 reference by 565 and data bit 5 reference by 570, and
3. five bits, data bits 4 to 0, referenced by 575, 580, 585, 590, and 595, respectively.

The five bits hold a 5 bit code selected from the codes 310 (Fig 3A) in the encode table 300 (Fig 3A).

The preferred embodiment of this invention uses the high order bit of the code byte 500 as the flag bit 510, because it results in the faster execution of the process. However, any bit could have been designated as the flag bit 510 with the same logical result.

Fig 6–Encoded Data Stream

Fig 6 shows a series of decimal values 610 comprising a first value 620 equal to decimal 0, a second value 622 equal to 0, a third value 624 equal to 0, a fourth value 626 equal to 0, a fifth value 628 equal to 0, a sixth value 630 equal to 2, and a seventh value 632 equal to 10. After the run-length encoding step 130 (Fig 1), the corresponding encoded data 140 (Fig 1) would be compressed down to four bytes of binary code 640 comprising a first byte 650 containing a repeat count 651, a second byte 652 containing a first code 653, a third byte 654 containing a second code 655, and a fourth byte 656 containing a third code 657. The repeat count 651 has a binary value of "0000101" which equals decimal five representing the run-length of the repeating value in the first five of the decimal values 610. The first code 653 has a binary value of "00000" which equals the repeated decimal value zero. The second code 655 has a binary value of "00010" which equals the non-repeated decimal value two. The third code 657 has a binary value of "01010" which equals the non-repeated decimal value ten.

Fig 7–Decode Table

Fig 7 illustrates a decode table 700 of the preferred embodiment of the present invention. The decode table 700 may be implemented in the C programming language, as shown, or any programming language, and is an array of 32 element with each element being a 32 bit pixel value 200 (Fig 2A). The decode table is comprised of a column of alpha values 710, a column of red values 720, and a column of green values 730, and a column of blue values 740, where the alpha values 710 are shifted by 24 bits, the red values 720 are shifted by 16 bits, and the green values 730 are shifted by 8 bits leaving the blue values 740 in place. The line in the decode table 700 contains one element. Each element comprising:

1. the alpha channel 208 (Fig 2A) with full intensity represented by hexadecimal 0xFF
2. the red channel 206 (Fig 2A)
3. the green channel 204 (Fig 2A)
4. the blue channel 202 (Fig 2A)

where the values of the three color channels are equal to the corresponding stepped values 360 (Figs 3A and 3B) associated with each line of the encode table 300 (Fig 3A).

Although each element is documented as an expression composed of various bit shift and bit-wise OR operations, the expression is evaluated by the compiler when the program is compiled so that each element of the decode table 700 is a 32 bit pixel value 200 (Fig 3A) ready for direct placement in the decompressed image.

In the preferred embodiment of this invention the 32 bit pixel value 200 (Fig 2A) is used; however other embodiments use the 24 bit pixel value 210 (Fig 2B) or other pixel sizes known in the art. Embodiments of other common pixel depths of 16 bit, 15 bit, 8 bit, 4 bit, 3 bit or 1 bit could be used without limiting the scope of this invention. Of course any source resolution less than or equal to five bits would benefit from a modified and shortened encode and decode table or could be encoded by shifting bits as explained below.

Fig 8–Alternate Code Selection

Fig 8 illustrates an alternate embodiment of this invention where tables are not used for encoding and decoding. Instead, the high order 5 bits 810-818 of an 8 bit pixel 800 are shifted to the right by 3 bit positions to form a five bit sample 830. The upper 3 bits of 830 are ignored bits 850. This shifting to obtain a five bit code replaces steps 405 and 406 of the flowchart in Fig 4A. The same run-length encoding method is used. During decompression the five bit sample 830 is shifted to the left by 3 bit positions to form a decompressed pixel 860, where the 3 low order bits 880 are filled with zero binary values.

Fig 9–Decode Flowchart

Fig 9 illustrates the decode flowchart which presents the details of the preferred embodiment of the value lookup step 170 (Fig 1) and the image reconstitution step 180 (Fig 1).

The decoding begins at a decode entry 900. In a decode initialization step 901, a repeat counter C is set to one, an encoded length L is set to the value obtained with the encoded data 140 (Fig 1), and an index I is set to 0. Next, a get code step 902 obtains a signed byte X from the encoded data 140 (Fig 1) array A. A determine type 906 decision checks to see if the signed byte X is less than 0.

If the signed byte X is less than zero, it is because the high order bit, the flag bit 510 (Fig 5A) is set to binary value 1, as in Fig 5B, indicating that the byte X is a repeat code 520. Flow goes to assign counter step 912 where the count 530 (Fig 5B) is extracted from X and placed in the repeat counter C and the next code is accessed by incrementing the index I and returning to the get code step 902.

If the signed byte X is greater than or equal to zero, it is because the flag bit 510 (Fig 5A) is set to binary 0, as in Fig 5C, indicating that the byte X is a data code 550. Flow goes to a decode lookup step 908 where the value of byte X is used to index into the decode table 700 (Fig 7) to obtain a pixel value V. Flow continues to a check zero count 909 decision.

The 909 decision always fails the first time ensuring that a place pixel step 910 is executed. The place pixel step 910 places the pixel value V in the next location of the decompressed image and decrements the repeat counter C and returns to the 909 decision. The pixel value V is placed repeatedly until C decrements to zero. Then the 909 decision branches flow to a reset counter step 914. At step 914 the repeat counter is reset to 1 and the index is incremented to select the next code.

Flow continues to the check length 916 decision where the index I is compared to the encoded length L to determine if there are more codes to be processed. If I is less than L flow returns to step 902, otherwise the decode process terminates at a decode exit 918.

The entire decode process is repeated for each frame image.

Figs 10A to 10C–Encrytpion Key, Encryption Table, and Decryption Table

Fig 10A shows an encryption key 1000. The first column shows the original code. The second column shows the corresponding cipher code.

Fig 10B shows an encryption table 1010, and Fig 10C shows a decryption table 1020. The encryption table 1010 has the same format as the standard encode table 300 (Fig 3A), and the decryption table 1020 has the same format as the decode table 700 (Fig 7). However the entries in both tables are rearranged such that the direct correlation between the intensity level and the position in the table is broken. When these versions of the tables are used, the encode and decode processes and their speed of execution are substantially the same but the encoded data 140 (Fig 1) becomes a cipher and has a higher level of security. It should be recognized by one with ordinarily skill in the art that there are other embodiments of the present invention with different encryption/decryption table rearrangements.

Advantages

Filtering and Image Enhancement

The stepped values 360 (Fig 3A) are a significant discovery of the present invention. The use of these specific values results in high quality decompressed images when the original image is generated by an electronic sensing device such as an ultrasound machine. The encode table 300 is arranged such that the spikes in the video signal are filtered in the high and low end, line numbers 31 and 0, respectively. The remaining values are distributed more evenly with larger ranges at line 3, 7, 15, 19, 23, and 27.

By altering the contents of the encode table 300 and the decode table 700 various filters can be implemented to enhance the image quality. A high or low noise filter can be beneficial when the image is generated by an imaging technology such as radar, ultrasound, x-ray, magnetic resonance,

or similar technology. Variations in the encode and decode table can be made to enhance the perceived quality of the decompressed image. Therefore, altering the contents, shape, or size of the encode table 300 and the decode table 700 is anticipated by this invention and specific values in the tables should not be construed as limiting the scope of this invention.

Execution Speed

The preferred embodiment of this invention use a number of techniques to reduce the time required to compress and decompress the data.

The methods require only a single sequential pass through the data. Both the compression steps 100 and the decompression steps 150 access a pixel once and perform all calculations.

When selecting the 8 bit pixel value 299, the preferred embodiment selects the low order bits from the 32 bit pixel value 200 or the 24 bit pixel value 210 so that an additional shift operation is avoided.

The encode table 300 is a fast and efficient way to convert the 8 bit pixel value 299 into one of the 5 bit codes 310.

The decode table 700 contains 32 entries each comprised of the 32 bit pixel value 200 that are ready for placement in the decompressed image. Although each element is documented as an expression composed of various bit shift and bit-wise OR operations, the expression may also be evaluated by a compiler when the program is compiled so that each element of the decode table 700 becomes a 32 bit pixel value 200.

General Purpose

Although the preferred embodiment of the present invention is tuned to the characteristics of a medical image, its lossless compression of the sampled data results in high quality video streams that have general purpose application in a number of areas including, without limitation, video conferencing, surveillance, manufacturing, and rich media advertising.

Lossless Nature/No Artifacts

Once the analog signal is sub-sampled and filtered to select a five bit code value which eliminates some of the real world defects, the methods of the present invention compress and decompress the data with no irreversible data loss. Unlike JPEG and MPEG, the decompressed image never suffers from artificially induced blocking or smearing or other artifacts that are result of the lossy compression algorithm itself. As a result even a small sub-sample of the image remains clear and true to the perceived quality of the original image.

Conclusion, Ramification, and Scope

Accordingly, the reader will see that the compression and decompression steps of the present invention provides a means of digitally compressing a video signal in real time, communicating the encoded data stream over a transmission channel, and decoding each frame and displaying the decompressed video frames in real time.

Furthermore, the present invention has additional advantages in that:

1. it provides a means of filtering real world defects from the video image and enhancing the image quality;
2. it allows for execution of both the compression and decompression steps using software running on commonly available computers without special compression or decompression hardware;
3. it provides decompressed images that have high spatial quality that are not distorted by artifacts of the compression algorithms being used;
4. is provides a scalable means of video compression; and
5. it provides a means for reducing the space required in a storage medium.

Although the descriptions above contain many specifics, these should not be construed as limiting the scope of the invention but as merely providing illustrations of some of the preferred embodiments of this invention. For example, stepped values in the encode and decode tables can be altered and the same relative operation, relative performance, and relative perceived image quality will result. Also, these processes can each be implemented as a hardware apparatus that will improve the performance significantly.

Thus the scope of the invention should be determined by the appended claims and their legal equivalents, and not solely by the examples given.